Priebeh krajského kola

Krajské kolo 39. ročníka Olympiády v informatike, kategória B, sa koná 23. 1. 2024 v dopoludňajších hodinách. Na riešenie úloh majú súťažiaci **4 hodiny čistého času**. Rôzne úlohy riešia súťažiaci na samostatné listy papiera. Akékoľvek pomôcky okrem písacích potrieb (napr. knihy, výpisy programov, kalkulačky) sú zakázané.

Čo má obsahovať riešenie úlohy?

- Slovne popíšte algoritmus.
  Slovný popis riešenia musí byť jasný a zrozumiteľný i bez nahliadnutia do samotného algoritmu/programu.
- Zdôvodnite správnosť vášho algoritmu.
- Uveďte a zdôvodnite jeho časovú a pamäťovú zložitosť.
- Podrobne uveďte dôležité časti algoritmu, ideálne vo forme programu v nejakom bežnom programovacom jazyku (napr. C++, Python, Java, Pascal).
- V prípade, že používate vo svojom programovacom jazyku knižnice, ktoré obsahujú implementované dátové štruktúry a algoritmy (napr. STL pre C++), v popise algoritmu stručne vysvetlite, ako by ste napísali program s rovnakou časovou zložitosťou bez použitia knižnice.

Hodnotenie riešení

Za každú úlohu môžete získať od 0 do 10 bodov.

Pokiaľ nie je v zadaní povedané ináč, najdôležitejšie dve kritériá hodnotenia sú v prvom rade **správnosť** a v druhom rade **efektívnosť** navrhnutého algoritmu. Na výslednom počte bodov sa môže prejaviť aj kvalita popisu riešenia a zdôvodnenie tvrdení o jeho správnosti a efektívnosti.

Efektívnosť algoritmu posudzujeme vypočítaním jeho časovej zložitosti – funkcie, ktorá hovorí, ako dlho vykonanie algoritmu trvá v závislosti od veľkosti vstupných parametrov. Nezávisí pri tom na konštantných faktoroch, len na rádovej rýchlosti rastu tejto funkcie.

V zadaní úloh uvádzame časť „Hodnotenie“, v ktorej nájdete približné limity na veľkosť vstupných údajov. Pod pojmom „efektívne vyriešiť“ chápeme to, že váš program spustený na modernom počítači by mal dať odpoveď nanajvýš do niekoľkých sekúnd.

Údaje z tejto časti zadania by mali slúžiť hlavne na to, aby ste o riešení, ktoré vymyslíte, vedeli približne povedať, koľko bodov zaň dostanete.

B-II-1 Kopa kníh 2

Knihovníčke sa robotické rameno pokazilo. Nuda ju však neopúšťa, a tak si z času na čas kopne do niektorej kopy kníh. To má za následok, že sa z vrchných n kníh tejto kopy stane nová kopa. V knižnici sa po novom dejú nasledovné tri typy udalostí:

1. Do knižnice sa vráti požičaná kniha x a knihovníčka ju položí na vrch niektorej kopy k.
2. Knihovníčka vezme vrchnú knihu z kopy k a vráti ju na správne miesto v knižnici.
3. Knihovníčka kopne do kopy k a vznikne tak nová kopa s novým identifikačným číslom k_2.

Knihovníčka má konzistentnú techniku kopu, takže číslo n je pri všetkých kopoch rovnaké. Ak by na kope, do ktorej kopla, bolo menej ako n kníh, tak sa všetky knihy presunú na novú kopu a tá pôvodná ostane prázdna. Čo najefektívnejšie simulujte uvedené tri typy udalostí.

Formát vstupu a výstupu

V prvom riadku vstupu je číslo n: veľkosť efektu kopanca. Nasleduje neznámy počet riadkov popisujúcich udalosti v poradí, v akom nastali. Každá udalosť je popísaná niekoľkými číslami, pričom prvé z nich určuje jej typ.

- Pre udalosti typu 1 (nová kniha číslo x pribudne na vrch kopy k) má riadok vstupu formát „1 x k“.
- Pre udalosti typu 2 (odoberieme vrchnú knihu kopy k) sa používa formát „2 k“.
- Pre udalosti typu 3 (kopneme do kopy k a vznikne tak kopa k_2) sa používa formát „3 k k_2“.
- Riadok „0“ signalizuje koniec vstupu (žiadne ďalšie udalosti).

Čísla x nemusia byť navzájom rôzne. Ak sa číslo k odkazuje na novú kopu, predpokladajte, že táto kopa je prázdna. Číslo k_2 sa vždy bude odkazovať na novú kopu, teda na takú, ktorá sa dovtedy na vstupe nezjavila.

Pre každú udalosť typu 2 vypíšte číslo knihy, ktorá bolo odobraná z vrchu kopy, alebo -1 ak bola kopa prázdna.

Obmedzenia a hodnotenie

V programe môžete predpokladať, že platí $0 \leq x \leq 10^9$ a $0 \leq k, k_2 \leq 10^6$.
Plný počet bodov viete dostať za riešenie, ktoré pre ľubovoľné n zvládne efektívne spracovať ľubovoľných 10^6 udalostí. Za správne riešenie, ktoré to zvládne za predpokladu $n \leq 10$, viete dostať aspoň 4 body.

Príklad

vstup

```
3
1 100 0
1 101 0
1 102 0
1 103 0
1 104 0
3 0 10
2 0
2 0
2 0
2 10
2 10
1 105 10
1 106 10
1 107 10
1 108 10
3 10 20
2 10
2 10
2 10
2 20
0
```

výstup

```
101
100
-1
104
103
105
102
-1
108
```

Po prvom kopanci sú na kope 0 *zdola hore knihy* $[100, 101]$ *a na kope* 10 *knihy* $[102, 103, 104]$.

Následne knihovníčka postupne vyprázdni kopu 0 *a odoberie časť kopy* 10. *Na tej ostane* $[102]$.

Potom prídu ďalšie knihy na kopu 10. *Po ich pridaní bude vyzerať nasledovne:* $[102, 105, 106, 107, 108]$.

Druhý kopanec ju rozdelí na kopy $[102, 105]$ *a* $[106, 107, 108]$.

Na záver knihovníčka postupne vyprázdni kopu 10 *a z kopy* 20 *odoberie vrchnú knihu.*

B-II-2 Umelecká záhrada

Peťo bol návštevou zrekonštruovanej SNG očarený natoľko, že sa rozhodol spraviť si zo svojej *kruhovej záhrady* kúsok umenia. Začal tým, že po jej obvode rovnomerne pozapichoval n kolíkov, ktoré si v smere hodinových ručičiek očísloval od 1 po n. Medzi tieto kolíky by rád natiahol farebné šnúry. Aby to bolo dostatočne umelecké, vybral si postupnosť kolíkov $a_1, a_2, \ldots, a_k$. Pre každé $i < n$ medzi kolíky a_i a a_{i+1} natiahne *rovnú šnúru*. (Nejde o cyklus: prvý a posledný kolík postupnosti nebudú prepojené šnúrou.)
Už pri prvej postupnosti však zistil, že má problém. Občas sa mu totiž stane, že sa niektoré šnúry križujú. A vtedy sa zle naťahujú, zamotávajú sa do seba a celkovo, nevyzerá to veľmi esteticky. Vyskúšal preto ďalšiu postupnosť, avšak s podobným výsledkom. Overovať každý svoj umelecký návrh naťahovaním šnúr je veľmi neefektívne, preto by ho zaujímalo, ako vie z navrhnutej postupnosti zistiť, či je to návrh krásneho umeleckého diela bez kríženia, alebo len ďalšia kolízna zbytočnosť.

Súťažná úloha

K dispozícii máte počet kolíkov $n \geq 3$ na obvode Peťovej záhradky a postupnosť $a_1, a_2, \ldots, a_k$ podľa ktorej chce Peťo naťahať šnúry. Zistite, či táto postupnosť popisuje také natiahnutie šnúr, v ktorej sa žiadne dve šnúry nekrižujú.

Formát vstupu a výstupu

Na prvom riadku sú celé dve kladné čísla n a k – počet kolíkov a dĺžka postupnosti.
Na druhom riadku je k medzerou oddelených čísel $a_1, a_2, \ldots, a_k$ – čísla kolíkov.
Platí, že $1 \leq a_i \leq n$ a všetky čísla tejto postupnosti sú navzájom rôzne.

Ak pri natiahnutí šnúr podľa danej postupnosti vznikne nejaké kríženie, vypíšte na výstup slovo `umeNIE`.
V opačnom prípade vypíšte slovo `umeÁNO`.

Obmedzenia a hodnotenie

Na plný počet bodov treba riešenie, ktoré efektívne vyrieši ľubovoľný vstup, v ktorom $n \leq 10^9$ a $k \leq 10^6$.

Na 8 bodov treba riešenie, ktoré efektívne vyrieši ľubovoľný vstup, v ktorom $n \leq 10^6$ a $\boldsymbol{k = n}$.
Inými slovami, na zisk 8 bodov stačí program, ktorý dostatočne efektívne vyrieši **ľahšiu verziu** plnej súťažnej úlohy: môžete predpokladať, že Peťova šnúra povedie postupne cez úplne všetky kolíky.
Ak riešite túto verziu súťažnej úlohy, **výrazne** to na začiatku riešenia uveďte.

Na 5 bodov treba riešenie, ktoré efektívne vyrieši ľubovoľný vstup, v ktorom $k \leq n \leq 5000$.

Za ľubovoľné správne riešenie môžete získať aspoň 3 body.

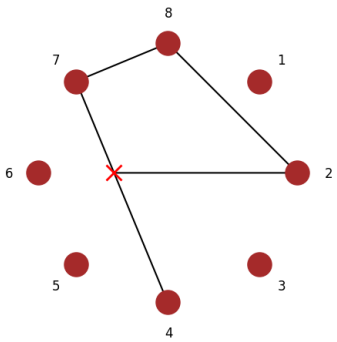
Príklad

vstup

```
8 5
4 7 8 2 6
```

výstup

```
umeNIE
```



B-II-3 Sklenené guľôčky

Izák našiel na povale škatuľu plnú sklenených guľôčok po prababičke. Z priloženého listu sa dočítal, že tieto guľôčky sú z mnohých druhov skla, a že z každého druhu skla sú v škatuli práve dve guľôčky. Jeho netrénovanému oku však všetky vyzerajú rovnako.
Pod škatuľou s guľôčkami našťastie našiel aj druhú škatuľu, v ktorej bol starý spektskloskop – prístroj na meranie druhov skla pomocou svetelného lúča. Spektskloskop dokáže pre ľubovoľnú skupinu guľôčok povedať, koľko rôznych druhov skla sa medzi nimi nachádza.
Izák by chcel rozdeliť guľôčky do dvojíc s rovnakým materiálom, ale má obavy, koľkokrát môže tento starý spektskloskop použiť, kým sa pokazí. Chcel by ho teda použiť čo najmenejkrát.

Úloha

Máme 2000 guľôčok, ktoré sú z 1000 na pohľad nerozoznateľných druhov skla, vždy po dve guľôčky z jedného druhu. Guľôčky si Izák označil centrofixkou číslami 1 až 2000, aby sa mu nepoplietli.
Izák môže zobrať ľubovoľnú množinu guľôčok, nasypať ich do spektskloskopu a ten mu po stlačení gombíka na displeji ukáže, koľko rôznych druhov skla sa nachádza v prístroji (a následne všetky guľôčky vysype von).

- Navrhnite postup (algoritmus), pomocou ktorého dokáže Izák zaručene nájsť všetky dvojice guľôčok.
- **Napíšte program alebo detailný pseudokód** tohto algoritmu.

  Detaily k písaniu programu nájdete nižšie v časti „Používanie spektskloskopu a formát výstupu“.
- Odhadnite, koľkokrát bude spektskloskop použitý v najhoršom možnom prípade. Nemusíte to vyčísliť presne, stačí na úrovni, „menej ako 1 000 000“, „menej ako 25 000“ a podobne. Čím menší (rádovo) odhad budete mať, tým viac bodov viete získať.

  Detaily nájdete v časti „Hodnotenie“.
- Zdôvodnite, že váš algoritmus zaručene vždy nájde správne riešenie na menej ako odhadnutý počet použití prístroja.

Používanie spektskloskopu a formát výstupu

Ak budete písať pseudokód vašetho algoritmu, predstavte si, že máte k dispozícii funkciu
`odmeraj(zoznam čísel) -> počet`,
ktorá ako jediný argument dostane zoznam čísel guľôčok (napr. `vector<int>` v C++, alebo `list`, resp. `List[int]` v Pythone) a na výstup vráti jedno číslo od 0 po 1000.
Vašou úlohou je niekoľkokrát zavolať túto funkciu a následne vypísať 2000 čísel oddelených medzerami na jeden riadok. Pre každé i má i-tym z týchto čísel byť číslo tej guľôčky, ktorá tvorí pár s guľôčkou i. Samozrejme musí platiť, že ak a-te číslo vo výstupe je b, tak b-te číslo musí byť a.

Príklad

Keby sme mali len štyri guľôčky, jedno možné riešenie úlohy by mohlo spektskloskop používať takto:

```
odmeraj( [1, 2, 4] ) -> 2
odmeraj( [1, 2] ) -> 2
odmeraj( [3, 2] ) -> 2
výstup: 3 4 1 2
```

Pri prvej otázke sme vložili do spektskloskopu guľôčky s číslami 1, 2 a 4 a spektskloskop nám dal odpoveď, že sú z dvoch druhov skla. (Tým sme sa vlastne nič nové nedozvedeli.)

Po odpovedi na druhú otázku vieme, že guľôčky 1 a 2 nie sú z rovnakého skla. To nám ale ešte nestačí.

Po tretej otázke vieme, že guľôčky 3 a 2 tiež nie sú z rovnakého skla. Tým pádom musí byť guľôčka 2 z rovnakého skla ako guľôčka 4, a teda nutne druhý pár tvoria guľôčky 1 a 3.

Iné riešenie by pre iné štyri guľôčky mohlo mať viac šťastia:

```
odmeraj( [1, 2] ) -> 1
výstup: 2 1 4 3
```

Už po prvej otázke vieme, že guľôčka 1 a 2 sú z rovnakého skla a tým pádom aj 3 a 4 musia byť rovnaké.

Dôležité upozornenie: Vašou úlohou nie je dopredu rozhodnúť, ako budú vyzerať všetky použitia spektskloskopu, ale navrhnúť algoritmus, ktorý sa bude **postupne** rozhodovať, ako spektskloskop používať. V závislosti od toho, aké odpovede algoritmus dostane, môže meniť svoje správanie.
Môžeme to vidieť aj na druhom príklade. Keď sme dostali odpoveď 1, môžeme rovno vypísať riešenie. Ak by sme na tú istú otázku dostali odpoveď 2, potrebovali by sme ešte aspoň raz použiť spektskloskop.
Vašou úlohou je navrhnúť algoritmus tak, aby v najhoršom prípade použil spektskloskop čo najmenejkrát.

Hodnotenie

- až 2 body môžete dostať, ak váš algoritmus nikdy nepoužije spektskloskop viac než 2 000 000-krát.
- až 4 body môžete dostať, ak váš algoritmus nikdy nepoužije spektskloskop viac než 1 000 000-krát.
- až 7 bodov môžete dostať, ak váš algoritmus nikdy nepoužije spektskloskop viac než 100 000-krát.
- až 10 bodov môžete dostať, ak váš algoritmus nikdy nepoužije spektskloskop viac než 25 000-krát.

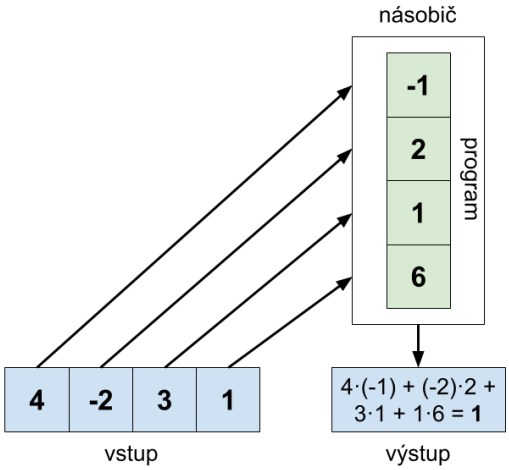
Pozor, pri hodnotení budeme klásť veľký dôraz na **dobre napísaný program alebo detailný pseudokód** a zdôvodnenie správnosti.

B-II-4 Tabuľkový počítač 2

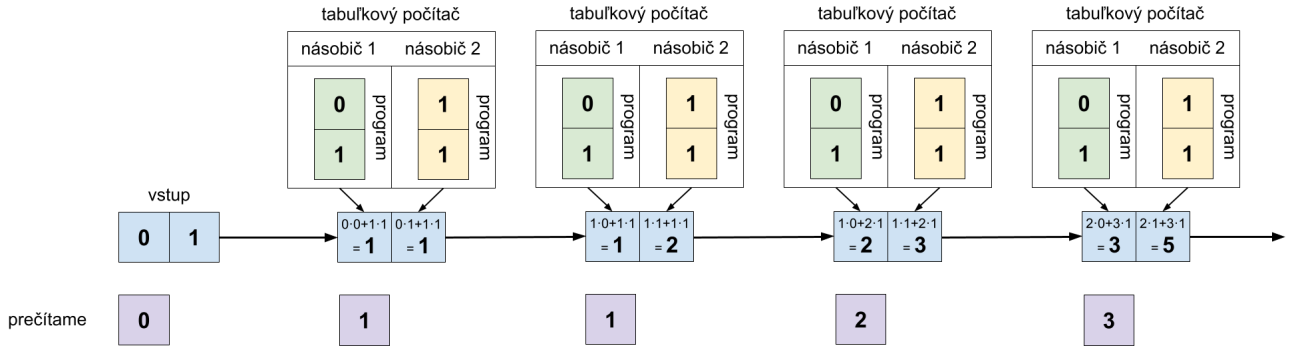
*Text tohto zadania je **totožný** s textom z domáceho kola, len časť Súťažná úloha je nová.*

Počas prázdnin u starých rodičov našiel Krištof na povale zvláštny mechanicky stroj, okolo ktorého sa povaľovalo množstvo pásikov tvrdého papiera. Opýtal sa starkého, na čo ten stroj slúži, a starký mu prezradil, že je to jeho starý *tabuľkový počítač* a ponúkol sa Krištofovi, že mu vysvetlí ako funguje.

Ako prvé mu ukázal papierové kartičky, ktoré sa volali *pásiky*. Pásik je úzky kúsok tvrdého papieru rozmerov $1 \times k$ rozdelený na k štvorčekov, do ktorých sa vpisujú čísla. Starký mal pásikov dostatočné množstvo pre všetky možné hodnoty k.
Následne starký Krištofovi predstavil hlavnú súčiastku tabuľkového počítača – *násobič*. Aby násobič fungoval, treba doň najprv vložiť *program*. Program je pásik dĺžky k, na ktorého každé políčko sa vpíše jedno číslo. Keď má násobič pripravený program, môžeme doň vložiť *vstup*. Vstup pre násobič je opäť pásik dĺžky k, do ktorého štvorčekov sú vpísané čísla.
Po vložení vstupu násobič „vynásobí“ vstup s programom a výslednú hodnotu, ktorú bude tvoriť *jedno číslo*, vypíše ako *výstup*. Počítanie výstupu prebieha tak, že násobič postupne vynásobí prvé číslo vstupu s prvým číslom programu, druhé číslo vstupu s druhým číslom programu, a tak ďalej až po k-te číslo. Takto získaných k hodnôt následne sčíta a tento súčet je výstupom násobiča.
Program je v násobiči pevne daný, je preto možné tento násobič použiť opakovane pre rôzne vstupy.
Keď bol starký v Krištofovom veku, veľa sa s násobičmi hral, vždy mu ale prekážalo, že z nich nevie naskladať žiaden zložitejší program. Vstupom pre násobič je totiž pásik s k číslami, zatiaľ čo jeho výstupom je len jedno číslo a s tým už násobič nevie ďalej nič robiť. Jedného dňa však starký prišiel na nápad, ako tento problém vyriešiť, a tak vznikol *tabuľkový počítač*, ktorý vedel po vložení jedného vstupu počítať donekonečna.
Starkého geniálna myšlienka bola vlastne veľmi jednoduchá: zobral k násobičov a uložil ich do radu. Potom do každého z nich vložil nejaký (nie nutne ten istý) program dĺžky k. (Tento výtvor nazval *tabuľkový počítač*, keďže na k programov dĺžky k sa môžeme dívať ako na tabuľku $k \times k$ čísel.)



Vstupom pre takýto tabuľkový počítač bol pásik s k číslami. Tento vstup sa zadal ako vstup každému z násobičov. Každý násobič tento spoločný vstup vynásobil so svojím konkrétnym programom a tým vyrobil ako svoj výstup jedno číslo. Celkovým výsledkom výpočtu teda bolo dokopy k čísel: jedno z každého násobiča. Svoje výstupy jednotlivé násobiče zapísali na nový pásik: prvý násobič na jeho prvý štvorček, druhý násobič na druhý štvorček, atď. Pre vstup, ktorým bol pásik dĺžky k, sme takto dostali aj na výstupe opäť pásik dĺžky k.
Tento nový pásik, ktorý sme práve získali, je opäť platným vstupom pre tabuľkový počítač. Aby sme dostali stroj, ktorý beží donekonečna, stačilo už teda pridať len jednu súčiastku, ktorá vždy výstupný pásik zobrala a vložila ho späť do tabuľkového počítača ako jeho nasledujúci vstup.
Ako ale na takomto počítači počítať niečo užitočné, keď vlastne teraz žiaden výstup nemá? Aj s tým si starký poradil: pridal stroju čítačku s reproduktorom. Tá z každého **vstupu**, teda ešte **pred aplikovaním násobičov**, prečítala číslo v jeho **prvom** štvorčeku.



Krištof si všimol, že po spustení vlastne tabuľkový počítač pomocou čítačky *generuje nekonečnú postupnosť celých čísel*. A starký mu prezradil, že keď sa správne nastavia programy jednotlivých násobičov a prvý vstupný pásik, výsledné postupnosti vôbec nie sú náhodné.

Zoberme si napríklad Fibonacciho postupnosť. Jej prvé dva členy sú 0 a 1 a každý ďalší sa vypočíta ako súčet dvoch predchádzajúcich členov. Začiatok Fibonacciho postupnosti teda vyzerá nasledovne: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
Matematicky túto definíciu vieme zapísať nasledovne: Nech F_n je n-tý člen Fibonacciho postupnosti. Pre prvé dva členy platí $F_0 = 0$ a $F_1 = 1$. Následne pre všetky $n \geq 2$ platí $F_n = F_{n-1} + F_{n-2}$.
No a pozrime sa na obrázok vyššie. Pri nastavení prvého násobiča na program $(0, 1)$, druhého násobiča na program $(1, 1)$ a počiatočnom vstupe $(0, 1)$ tento tabuľkový počítač generuje práve Fibonacciho postupnosť.
Prečo je to tak? Vstup, z ktorého začíname, je $(0, 1)$. Ako vidíme z obrázku, po prvom použití násobičov dostaneme na výstupe pásik $(1, 1)$. Keď ten dáme na vstup a znova použijeme násobiče, dostaneme na výstupe

$(1, 2)$. Po treťom použití násobičov máme výstup $(2, 3)$ a po štvrtom $(3, 5)$. Ľahko si všimneme, že zatiaľ platí, že po n opakovaniach máme na prvom políčku pásiku hodnotu F_n a na druhom hodnotu F_{n+1}.
Bude toto naozaj platiť donekonečna? Áno: keď dáme tabuľkovému počítaču na vstupe pásik (F_n, F_{n+1}), tak prvý násobič vypočíta hodnotu $0 \cdot F_n + 1 \cdot F_{n+1} = F_{n+1}$ a druhý vypočíta hodnotu $1 \cdot F_n + 1 \cdot F_{n+1} = F_{n+2}$. Ďalším vstupom bude preto pásik (F_{n+1}, F_{n+2}) – teda opäť dve po sebe idúce Fibonacciho čísla, akurát s o 1 väčším indexom.

Takýto postup sa dá využiť aj pri iných postupnostiach, stačí si správne zvoliť konštantu k a potom správne naprogramovať jednotlivé násobiče. Ukážeme si ešte jeden trochu zložitejší príklad.
Majme postupnosť T, kde $T_0 = 4$, $T_1 = -2$, $T_2 = 3$ a pre $n \geq 3$ platí $T_n = T_{n-1} + 2 \cdot T_{n-2} - 7 \cdot T_{n-3} + 6$.
Takúto postupnosť by generoval napríklad tabuľkový počítač s násobičmi $(0, 1, 0, 0)$, $(0, 0, 1, 0)$, $(-7, 2, 1, 6)$ a $(0, 0, 0, 1)$ a vstupom $(4, -2, 3, 1)$. Skúste si vypočítať, ako budú vyzerať prvé pásiky, ktoré postupne bude vyrábať na výstupe, a rozmyslite si, prečo generuje práve vyššie definovanú postupnosť.
Všimnite si tiež, že rôzne tabuľkové počítače môžu generovať tú istú postupnosť. Ak by sme vo vyššie uvedenom príklade zmenili program tretieho násobiča na $(-7, 2, 1, 1)$ a hodnoty na vstupnom pásiku na $(4, -2, 3, 6)$, takýto tabuľkový počítač by tiež generoval postupnosť T.

Súťažná úloha

V tejto súťažnej úlohe dostanete popis niekoľkých postupností celých čísel. Vašou úlohou bude ku každej z nich navrhnúť tabuľkový počítač, ktorý ju bude generovať.

a) (2 body) Postupnosť A, ktorej prvky sú postupne: $0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1 \ldots$
Je to teda nekonečná postupnosť, v ktorej sa dookola opakujú čísla 0, 1, 2, 3 a 4.

b) (3 body) Postupnosť B, ktorej prvky sú postupne: $2^0, 3^0, 2^1, 3^1, 2^2, 3^2, 2^3 \ldots$
Je to teda nekonečná postupnosť, v ktorej sú nastriedačku čoraz vyššie mocniny čísel 2 a 3.
Začiatok tejto postupnosti: 1, 1, 2, 3, 4, 9, 8, 27, 16 ...

c) (2 body) Postupnosť C, kde $C_0 = 5$ a pre $n \geq 1$ platí $C_n = 4n + C_{n-1} + C_{n-2} + \cdots + C_1 + C_0$.
Hodnotu C_n teda získame ako súčet všetkých predchádzajúcich hodnôt postupnosti C, ku ktorým pripočítame $4n$.
Začiatok tejto postupnosti: 5, 9, 22, 48, 100, 204, ...

d) (3 body) Postupnosť D, kde $D_0 = -4$ a pre $n \geq 1$ platí
$D_n = 3 + 2 \cdot D_{n-1} - D_{n-2} + D_{n-3} - D_{n-4} + \cdots \pm D_0$.
Pri počítaní hodnoty D_n sa predchádzajúce prvky nastriedačku pričítavajú a odčítavajú. Keďže to, či pričítame alebo odčítame hodnotu D_0 záleží na parite čísla n, v našom vzorci je použitý symbol $\pm$ (pri párnych n bude D_0 odčítané, pri nepárnych pričítané). Neprehliadnite, že na začiatku je „$2 \cdot D_{n-1}$“.
Začiatok tejto postupnosti: -4, -5, -3, -2, 1, 5, 12, 23, 41 ...

Pri hodnotení vaších riešení bude záležať len na ich správnosti.
Konštantu k si v každej podúlohe môžete zvoliť ľubovoľne.

V riešení každej podúlohy odovzdajte nasledovné veci:

- Program tabuľkového počítača, teda k, čísla na prvom vstupe a programy jednotlivých násobičov.
- Slovný popis toho, ako ste tento program zostrojili, a zdôvodnenie, že váš tabuľkový počítač naozaj generuje zadanú postupnosť.

TRIDSIATY DEVIATY ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Truc Lam Bui, Ján Hozza, Timea Szöllősová
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: NIVAM – Národný inštitút vzdelávania a mládeže, Bratislava 2024